Data Science Case - Topic Extraction

Submitted by: Novojit Saha

After reading about the case, I researched on how to extract topics from documents. I figured this is called “Topic Modelling”, which belongs to the field of NLP.
I decided to use Jupyter Notebooks because it is easy to visualise the process and the results there.

I began with implementing LDA technique, just to get a feel of the problem at hand. However, it’s a very fundamental method and does not meet the requirements. LLMs do a great job at topic modelling, but they’re not the most scalable solution out there.
So I tried BERTopic: a topic modelling framework (fig. 1). It comes with good documentation and flexibility.

Input Preprocessing (analysis.ipynb)
I selected the abstracts to extract information from. I found abstracts to provide the right balance between information and input size, compared to titles and claims texts.
Next step was to calculate descriptive statistics on the character length of abstracts to understand the input data (*Appendix A*). I found that some abstracts were empty and/or have really small lengths. I define abstracts with a character length <= 150 to be insufficient. I replaced these with the corresponding claims text. After the replacement, there wasn’t any significant change in the shape of the dataset. However, the minimum length was 19 characters, and we will consider these patents as bad data.
Finally, I clean the abstract strings by removing newline characters (‘\n’) and HTML tags and export them to a CSV file, along with *ucid* and *claims*.

Embedding (embedding.ipynb)
BERTopic’s default pipeline uses the all-MiniLM-L6-v2 sentence transformer model. While it does a decent job at creating embeddings, I decided to experiment a little and use the stella_en_400M_v5 embedding model. I chose this model because, as of now, it is ranked 4th on the Massive Text Embedding Benchmark (MTEB) Clustering Category and is also not too big (400M parameters). It took about 5 minutes to encode the dataset in the hardware mentioned in Appendix A.
I pre-computed the embeddings and saved it to a CSV file for later use.

Dimensionality Reduction and Clustering
For these tasks, I kept the default BERTopic configurations. I studied these sub-tasks and didn’t find any important changes to make to fit my use case.

Vectorizer
This sub-task is responsible for creating the topic representations.

```
vec_model = CountVectorizer(min_df=2,stop_words='english',
ngram_range=(1,2))
```

The parameter *min_df* determines how frequent a word must be before being added to the representation. I selected 2 since I found this to provide ideal representations. The *stop_words* remove the stop words (e.g. is, to, from) from the representations

Fine-tuning

The topics generated by the default c-TF-IDF technique are not human readable/understandable. This is where we need help from LLMs. But, before passing the representations to the LLMs, I pass them through a Maximal Marginal Relevance (MMR) algorithm, which reduces redundancy and improves diversity of keywords. This ensures synonyms don't negatively affect the topic representations.

While I could've used OpenAI/Gemini's API to create the labels, I decided to try out the latest open-source small LLMs: Gemma-2-2B-Instruct and Llama-3.1-8B-Instruct models. These models were released around the last week of July, and have impressive benchmark scores. Both these models took around 45 minutes to generate around 450 labels in the hardware mentioned in the appendix. The prompt used is given in Appendix A..

Results

Changing the embedding from all-MiniLM-L6-v2 to stella_en_400M_v5 increases the number of clusters formed by a small amount, about 5%. About 50% of the documents are topic modelled, while the rest are modelled as outliers. These outliers can be fitted inside the clusters using BERTopic's outlier reduction methods, but I don't find that to be a good idea. The calculated approximate probabilities are very small and it feels forced to assign these documents to the discovered topics.
Despite spending most of the time implementing and optimising BERTopic, I haven't achieved satisfactory results.

Finally, I decided to use LLMs directly to topic label these documents. Once again, I tried the Gemma-2-2B-Instruct and Llama-3.1-8B-Instruct to topic label documents. Due to hardware limitations, I ran these models on a dataset of 50 documents. In my opinion, they do a great job at deciding the topics. However, only a domain expert can rate the quality of these topics.

To topic label 50 documents, Gemma takes ~1 second/document while Llama takes ~3 second/document in the given hardware (Appendix A). Moreover, Gemma sometimes fails to maintain a consistent output format, which makes it harder to process the outputs. On the other hand, Llama maintains a consistent output format.

Please note, the hardware I used is shared among other stakeholders, so I might not have gotten full capacity. It's not possible for me to check if the hardware is being used by others or not at any given time.

Based on what I learned so far, no methods can beat LLMs in terms of topic label quality, when it comes to topic modelling challenging documents. While scalability and cost might become an issue, such issues can be circumvented by using smaller open-source models.

I also made a video explaining the decision to use LLMs:
https://www.youtube.com/watch?v=f0nRUPHjbwE

References

- https://maartengr.github.io/BERTopic/index.html
- https://youtu.be/KZL7hkXSR1A?si=yXwNKJUxDecvqqRA

Appendix A

- **Initial abstract length descriptive stats**

```
count    46666.000000
mean       989.909549
std        381.994588
min          1.000000
25%        762.000000
50%        968.000000
75%       1174.000000
max      15462.000000
```

- **Abstract length descriptive stats after replacing insufficient abstracts with corresponding claims text.**

```
count     46666.000000
mean       1007.090001
std         729.553572
min          19.000000
25%         767.000000
50%         970.000000
75%        1176.000000
max      101791.000000
```

- **Hardware**
  - Dell PowerEdge R750
  - 72 Cores / 144 Threads / Max. 2.1 GHz
  - 256 GB Memory
  - 2 x Nvidia A16 (CUDA capable for GPU-calculations)

- **Prompt used to generate labels from BERTopic representations.**

  I've a topic that contains the following documents:
  [DOCUMENTS]

  The topic is described by the following keywords: '[KEYWORDS]'.

  Based on the information above, extract only one short topic label. Output only the topic label in the following format: topic: <topic label>

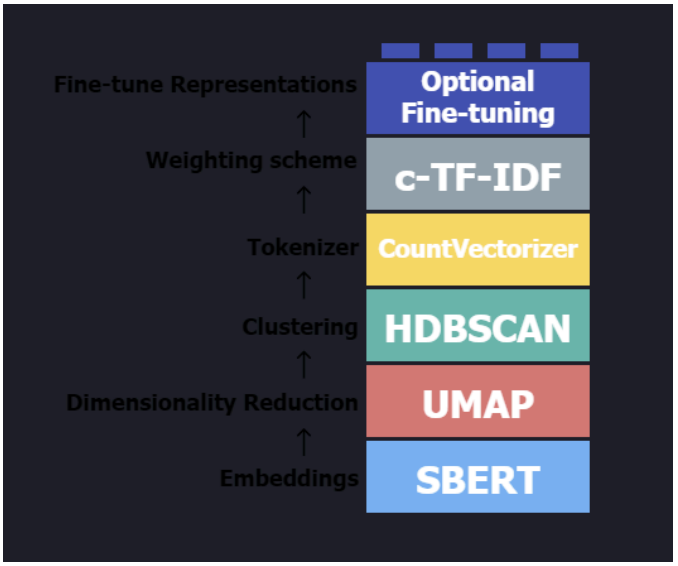
  [DOCUMENTS] is replaced by the top 3 most representative docs and *[KEYWORDS]* is replaced by the topic keywords.

- **Ideal project directory structure**

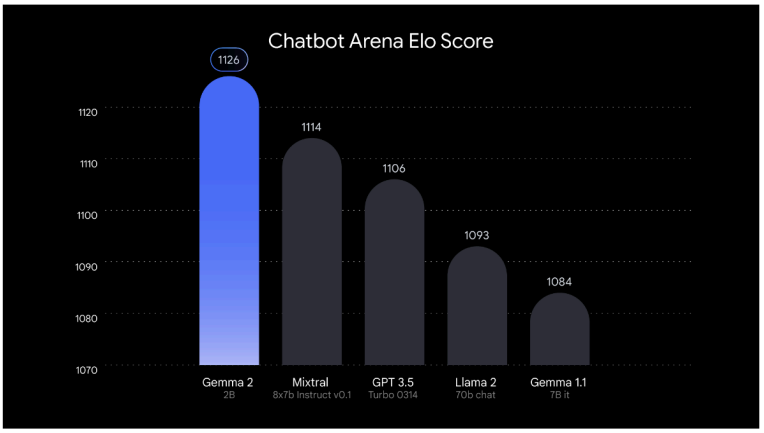
GETFOCUS_ASSIGNMENT
├── archive *(Contains experimental notebooks)*
│ ├── keybert.ipynb *(BERTopic with KeyBERTInspired representation)*
│ ├── lda.ipynb *(fundamental Latent Dirichlet Allocation implementation)*
│ └── mmr.ipynb *(BERTopic with MaximalMarginalRelevance representation)*
├── data *(not in github repo, requires manual creation)*
│ └── case_data.json
├── output_data *(not in github repo, requires manual creation)*
├── embeddings *(not in github repo, requires manual creation)*
│ ├── all-MiniLM-L6-v2.csv *(not in github repo, download from drive)*
│ └── stella_en_400M_v5.csv *(not in github repo, download from drive)*
├── venv *(python virtual env, not in github repo, requires manual creation)*
├── .gitignore
├── analysis.ipynb *(analysis of abstracts)*
├── bertopic_gemma_2.ipynb *(BERTopic with Gemma 2 for generating topic labels)*
├── bertopic_llama_3_1.ipynb *(BERTopic with Llama 3.1 for generating topic labels)*
├── bertopic.ipynb *(Generic BERTopic Implementation)*
├── embeddings.ipynb *(Pre-compute embeddings)*
├── gemma2_2b.ipynb *(Topic Labelling with Gemma 2)*
├── llama_3_1.ipynb *(Topic Labelling with Llama 3.1)*
├── preprocess_data.ipynb *(Extraction and cleaning of input data)*
└── README.md

Appendix B

- **Figure 1: BERTopic Overview**



- **Figure 2: Gemma-2-2B Comparison. Source:** https://developers.googleblog.com/en/smaller-safer-more-transparent-advancing-responsible-ai-with-gemma/



LMSYS Chatbot Arena leaderboard scores captured on July 30th, 2024. Gemma 2 2B score +/- 10.

- **Figure 3: Llama 3.1 8B Comparison. Source:** https://ai.meta.com/blog/meta-llama-3-1/

Category / Benchmark	Llama 3.1 8B	Gemma 2 9B IT	Mistral 7B Instruct	Llama 3.1 70B	Mixtral 8x22B Instruct	GPT 3.5 Turbo
General MMLU (0-shot, CoT)	73.0	72.3 (5-shot, non-CoT)	60.5	86.0	79.9	69.8
MMLU PRO (5-shot, CoT)	48.3	-	36.9	66.4	56.3	49.2
IFEval	80.4	73.6	57.6	87.5	72.7	69.9
Code HumanEval (0-shot)	72.6	54.3	40.2	80.5	75.6	68.0
MBPP EvalPlus (base) (0-shot)	72.8	71.7	49.5	86.0	78.6	82.0
Math GSM8K (8-shot, CoT)	84.5	76.7	53.2	95.1	88.2	81.6
MATH (0-shot, CoT)	51.9	44.3	13.0	68.0	54.1	43.1
Reasoning ARC Challenge (0-shot)	83.4	87.6	74.2	94.8	88.7	83.7
GPQA (0-shot, CoT)	32.8	-	28.8	46.7	33.3	30.8
Tool use BFCL	76.1	-	60.4	84.8	-	85.9
Nexus	38.5	30.0	24.7	56.7	48.5	37.2
Long context ZeroSCROLLS/QuALITY	81.0	-	-	90.5	-	-
InfiniteBench/En.MC	65.1	-	-	78.2	-	-
NIH/Multi-needle	98.8	-	-	97.5	-	-
Multilingual Multilingual MGSM (0-shot)	68.9	53.2	29.9	86.9	71.1	51.4